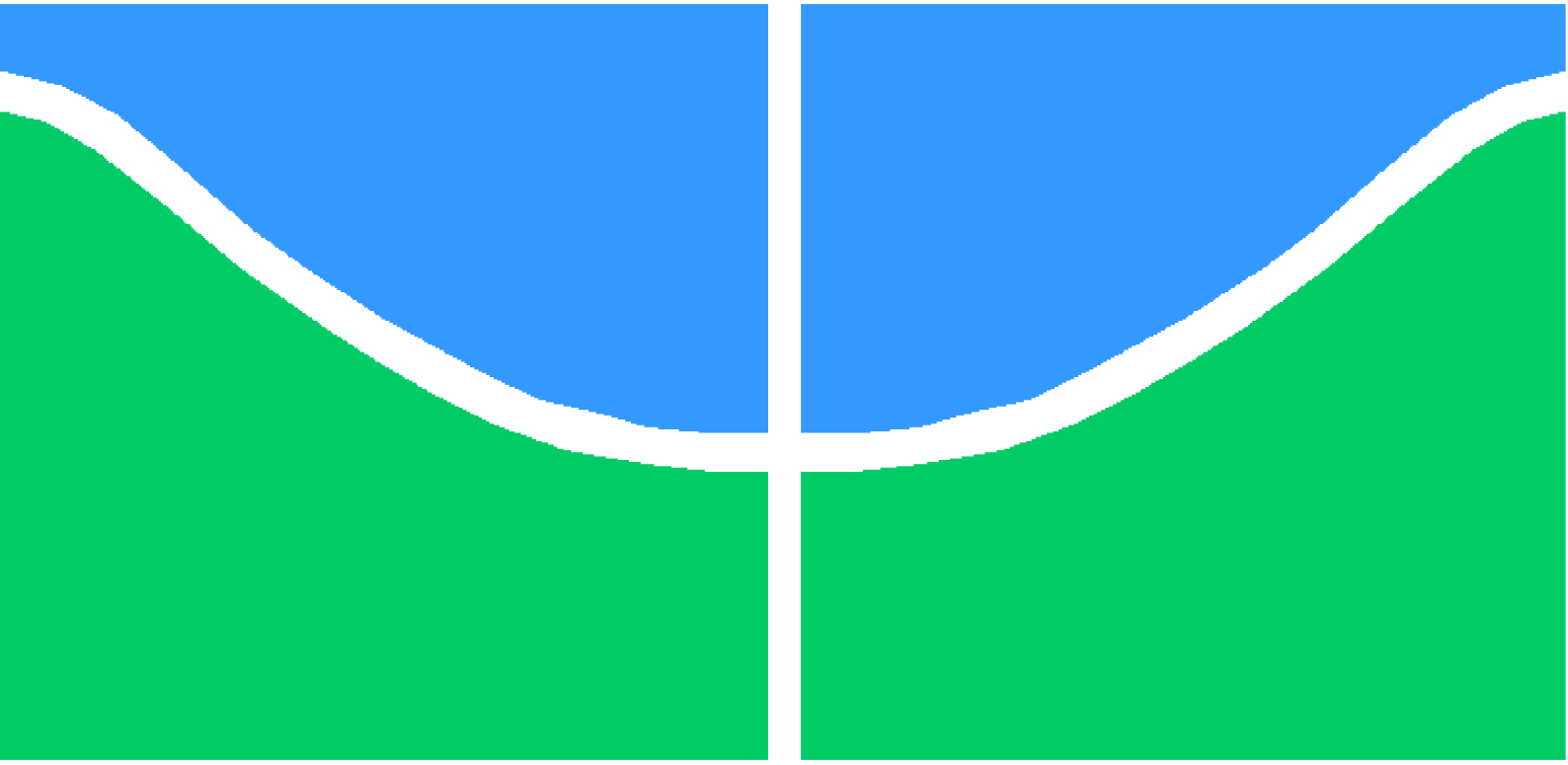

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Apoio a Promoção da Visibilidade da Dívida Técnica

Autor: Iago Rodrigues Gonçalves
Orientador: Dr. André Luiz Peron Martins Lanna

Brasília, DF
2018

Iago Rodrigues Gonçalves

Apoio a Promoção da Visibilidade da Dívida Técnica

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dr. André Luiz Peron Martins Lanna

Brasília, DF

2018

Iago Rodrigues Gonçalves

Apoio a Promoção da Visibilidade da Dívida Técnica/ Iago Rodrigues Gonçalves. – Brasília, DF, 2018-

51 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. André Luiz Peron Martins Lanna

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Dívida Técnica. 2. Metodologias Ágeis. I. Dr. André Luiz Peron Martins Lanna. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Apoio a Promoção da Visibilidade da Dívida Técnica

CDU 02:141:005.6

Iago Rodrigues Gonçalves

Apoio a Promoção da Visibilidade da Dívida Técnica

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 6 de julho de 2018:

Dr. André Luiz Peron Martins Lanna
Orientador

Msc. Cristiane Soares Ramos
Universidade de Brasília - UnB Faculdade Gama

Msc. Ricardo Ajax
Universidade de Brasília - UnB Faculdade Gama

Brasília, DF
2018

Agradecimentos

Primeiramente gostaria de agradecer a minha família, em especial meus pais, que sempre lutaram para que eu tivesse uma educação de qualidade, acreditando em meu potencial e confiando que essa educação abriria portas e oportunidades para mim. Sem eles eu não chegaria onde eu estou, com a visão que eu tenho.

Agradeço também aos meus amigos de coração, em especial a Gabriela e ao Rafael, os quais nutro tanto carinho em meu coração, de modo que para mim eles já são como família. Suas amizades sempre presentes moldaram e moldam meu caráter, valores e objetivos, além da segurança que me proporcionam, pois sei que poderei contar com eles para o resto da minha vida.

Resumo

A Dívida Técnica é uma metáfora que retrata a perda de qualidade no código, sendo utilizada para tomada de decisões durante o desenvolvimento. Essa dívida técnica parece estar mais presente nas metodologias ágeis de desenvolvimento de software devido a ênfase em entrega de funcionalidades, em que a dívida pode acabar prejudicando a produtividade do time a longo prazo, sendo interessante para essas metodologias poderem tomar decisões relacionadas a dívida técnica de modo mais estratégico para beneficiar a organização. Mas para isso é necessário que essa dívida esteja visível para as partes interessadas, sendo que a dívida precisa ser identificada, coletada e comunicada na organização.

Este trabalho discute sobre como é possível promover a visibilidade da Dívida Técnica dentro de um contexto de metodologias ágeis de desenvolvimento. Para isso foi identificado que técnicas existem para se identificar, medir e comunicar a dívida, e bem quais métricas existem para se estimar a dívida. Por fim, é proposto um método para promover a visibilidade, que permite a identificação, medição e comunicação da dívida, podendo fornecer os valores estimados em duas métricas diferentes, a de esforço e em valores financeiros.

Palavras-chaves: Dívida Técnica; Metodologias Ágeis; Gestão do Desenvolvimento de Software;

Abstract

The Technical Debt is a metaphor that portrays the loss of quality in the code, being used for decision making during development. This technical debt seems to be more present in the agile methodologies of software development due to the emphasis on functional delivery, in which the debt can end up harming the team's productivity in the long term, being interesting for these methodologies to be able to make decisions related to technical debt in a way to benefit the organization. But for this it is necessary for the debt to be visible to the interested parties, being that the debt needs to be identified, estimated and communicated in the organization.

This work discusses how it is possible to promote the visibility of the Technical Debt within a context of agile methodologies of development. For this, it was identified what techniques exist to identify, measure and communicate the debt, and well what metrics exist to estimate the debt. Finally, a method is proposed to promote visibility, which allows the identification, measurement and communication of the debt, and can provide the estimated values in two different metrics, effort and financial figures.

Keywords:Technical Debt; Agile Methodologies; Software Development Management;

Lista de ilustrações

Figura 1 – Quadrante da Dívida Técnica (Fonte: (FOWLER, 2009) adaptado) . . 28
Figura 2 – *Resultados da elicitação manual (fonte: (ZAZWORKA et al., 2013))* . 34
Figura 3 – *Code Christmas Tree* fonte: (KAISER; ROYSE, 2011) 35
Figura 4 – Estimativa de Dívida Técnica dada pela ferramenta SonarQube 39
Figura 5 – Ilustração da TD presente em um projeto através de um gráfico 39
Figura 6 – Representação do valor total da TD presente em um projeto 40
Figura 7 – Ilustração do método proposto (fonte: o autor) 42

Lista de tabelas

Tabela 1 – Conceitos e atributos relacionadas a TD (Adaptado de (OLIVEIRA, 2011) 27
Tabela 2 – Exemplo de Modelo de Qualidade SQALE 32
Tabela 3 – Exemplo de Modelo de Análise SQALE 32
Tabela 4 – Comparação das técnicas com os atributos de seleção 38

Lista de abreviaturas e siglas

TD	do inglês, *Technical Debt*
TDM	do inglês, *Technical Debt Management*
SQALE	do inglês, *Software Quality Assessment based on Lifecycle Expectations*
XP	do inglês, *Extreme Programming*
TCC	Trabalho de Conclusão de Curso

Sumário

I CONTEXTUALIZAÇÃO 19

1 INTRODUÇÃO . . . 21
1.1 Contextualização . . . 21
1.2 Descrição do Problema . . . 22
1.3 Objetivo . . . 22
1.4 Organização do Trabalho . . . 23

2 REFERENCIAL TEÓRICO . . . 25
2.1 Metodologias Ágeis . . . 25
2.2 Dívida Técnica . . . 27
2.2.1 Dívida Técnica nas Metodologias Ágeis de Desenvolvimento de Software . . . 30
2.2.2 Técnicas de Identificação, Medição e Comunicação da Dívida Técnica . . . 30
2.3 Considerações Finais . . . 35

3 MÉTODO DE PROMOÇÃO DA VISIBILIDADE E MÉTRICA DA DÍVIDA TÉCNICA . . . 37
3.1 Método e Métrica . . . 37
3.1.1 Técnicas para compor o método . . . 37
3.1.1.1 Seleção das técnicas . . . 37
3.1.1.2 Composição do SQALE com a estimativa através de fórmula . . . 38
3.1.2 Métricas utilizadas para medição da TD . . . 40
3.1.3 Comunicação da TD estimada . . . 41
3.1.4 Proposta de método . . . 41
3.2 Considerações Finais . . . 42

4 VALIDAÇÃO . . . 43
4.1 Validação do método . . . 43
4.1.1 Evidências Sobre a Promoção da Visibilidade . . . 43
4.1.2 Aplicação do Método para Validação . . . 44
4.2 Limitações . . . 44

5 CONSIDERAÇÕES FINAIS . . . 47

REFERÊNCIAS . . . 49

Parte I

Contextualização

1 Introdução

Este capítulo apresenta a contextualização sobre o tema abordado no trabalho, além da descrição do problema, os objetivos e a estrutura do restante do documento.

1.1 Contextualização

A Dívida Técnica (do inglês, *technical debt* - TD) é uma metáfora que diz que quando a qualidade do código é comprometida é como se assumíssemos uma dívida, em que essa dívida inicialmente acelera o desenvolvimento do software mas ela deve ser paga através da refatoração do código (CUNNINGHAM, 1992). Essa perda de qualidade pode ocorrer para atender alguma restrição, como tempo ou custo, ou para conquistar alguma oportunidade de mercado. O problema é que quando a dívida não é paga ela incorre "juros",que seria o aumento do custo para manter e evoluir o software no decorrer do tempo devido a presença da dívida (OLIVEIRA, 2011).

Com o passar dos anos a metáfora da TD foi ganhando mais atenção tanto da academia quanto da indústria devido à sua relevância no desenvolvimento de software (LI; AVGERIOU; LIANG, 2015). Também foram havendo interpretações diferentes da metáfora, em que o seu conceito, que era ligado inicialmente ao artefato de código, foi se estendendo a outros artefatos de desenvolvimento de software como arquitetura, design, teste, documentação, entre outros, não havendo um consenso de onde a metáfora se limita (LI; AVGERIOU; LIANG, 2015).

A metáfora vem ganhando interesse no contexto de desenvolvimento de software, em especial em metodologias ágeis que indicam estarem mais propensas à aquisição da TD (BEHUTIYE et al., 2017). Dentre os prejuízos identificados pela presença da dívida nos projetos de desenvolvimento que utilizam essas metodologias tem-se a perda de produtividade a longo prazo devido ao esforço empreendido para se consertar erros ou a dificuldade em implementar as novas funcionalidades (POWER, 2013), ou uma degradação da relação com o cliente (LIM; TAKSANDE; SEAMAN, 2012). Contudo, a TD pode ser utilizada de maneira a inicialmente acelerar o desenvolvimento e permitir entregas de funcionalidades importantes mais rapidamente e com isso ganhar um *feedback* do cliente, de modo a validar a necessidade do software a fim de evitar desperdício de recursos ou para conquistar clientes em um mercado competitivo (LIM; TAKSANDE; SEAMAN, 2012).

Assim, a utilização estratégica da TD pode trazer benefícios e promover a competitividade da empresa no mercado mas para isso deve-se conhecer os limites de tolerância de TD no código, além de coletar e comunicar a dívida presente para que a tomada de

decisões com relação a gestão da TD seja feita de forma mais objetiva.

1.2 Descrição do Problema

A falta de visibilidade da TD nas metodologias ágeis de desenvolvimento de software propicia sua aquisição e pode impedir sua mitigação (NORD et al., 2012), fazendo com que os projetos acumulem TD e sofram com os seus impactos. Os impactos mais relatados são a redução da produtividade, degradação da qualidade do sistema e aumento do custo de manutenção (BEHUTIYE et al., 2017). As metodologias ágeis aparentam estar mais propensas a adquirir TD (BEHUTIYE et al., 2017), sendo que uma das principais fontes de aquisição é a ênfase em entrega rápida de funcionalidades (TORKAR, 2011) em que o foco apenas na entrega de valor de produto pode acabar prejudicando aspectos de qualidade a longo prazo (NORD et al., 2012), ou em casos onde há pressão de tempo para entrega do software e o time acaba implementando uma solução de menor qualidade para conseguir uma entrega mais rápida (CODABUX; WILLIAMS, 2013).

Essa aquisição nem sempre é estratégica e pode ser negligenciada quando não há métricas que ilustrem a TD e os seus impactos no projeto. Os *stakeholders* que não são da área de desenvolvimento tendem a optar por adquirir dívida para poder acrescentar mais valor de negócio ao produto, além de serem mais resistentes a ideia de empreender esforços para saldar a TD (LIM; TAKSANDE; SEAMAN, 2012), pois não percebem os impactos negativos a longo prazo de uma má qualidade de código e enxergam as atividades de manutenção de qualidade como se apenas houvessem custos, sem possuir benefícios (NORD et al., 2012). Assim, a equipe não consegue se planejar para adquirir a TD em momentos estratégicos ou a saldá-la quando necessário. Em contraste a isso, há estudos que indicam que quando a TD está visível ela pode ser utilizada de maneira estratégica em benefício da organização (ALLMAN, 2012).

Assim, visando contribuir com uma solução para a falta de visibilidade da TD foi definida a seguinte questão de pesquisa:

- Como é possível apoiar a visibilidade a dívida técnica durante o desenvolvimento?

1.3 Objetivo

O objetivo geral desse trabalho é propor um método que apoie a promoção da visibilidade da TD durante o desenvolvimento. Através da promoção da visibilidade espera-se que o desenvolvimento possa equilibrar a entrega de funcionalidades e a manutenção da qualidade do código.

Esse objetivo foi detalhado em sub-objetivos, sendo eles:

- identificar as métricas existentes para a medição da TD;
- identificar técnicas que permitam a identificação, medição e comunicação da TD e,
- integrar as técnicas e métricas para compor uma estratégia que gere uma estimativa de TD em uma métrica adequada e que permita a sua visualização.

1.4 Organização do Trabalho

Os capítulos desse trabalho estão organizados da seguinte forma:

- **Capítulo 2 - Referencial Teórico**: capítulo onde são apresentados os fundamentos utilizados neste trabalho.
- **Capítulo 3 - Método de Promoção da Visibilidade e Métrica da Dívida Técnica**: capítulo onde é discutido como as técnicas e métricas encontradas podem ser integradas para compor a estratégia para dar visibilidade a TD.
- **Capítulo 4 - Validação**: capítulo onde é discutida a validação da solução apresentada, tendo em vista o objetivo do trabalho.
- **Capítulo 5 - Considerações Finais**: capítulo onde são realizadas as considerações finais do trabalho.

2 Referencial Teórico

Neste capítulo serão apresentados os fundamentos para o entendimento do tema abordado no trabalho. A subseção de Metodologias Ágeis traz uma introdução a essas metodologias, listando seus valores e princípios. A seção de Dívida Técnica explica do que se trata a metáfora e apresenta um breve visão do que existe na literatura sobre o assunto, o relacionamento da TD com as metodologias ágeis e técnicas existentes na literatura para a gestão da TD.

2.1 Metodologias Ágeis

As metodologias ágeis de desenvolvimento de software surgiram em resposta às críticas relacionadas às metodologias tradicionais de desenvolvimento de software. Essas críticas estavam centradas nas dificuldades das metodologias tradicionais em responder às mudanças e atender às expectativas dos clientes. Metodologias como a Cascata são conduzidas de forma linear, com o levantamento de requisitos, desenvolvimento, teste e entrega para o cliente, onde havia pouco espaço para negociação dos requisitos ou mesmo validação dos mesmos e *feedback* dos usuários. Com isso, foram surgindo metodologias iterativas e incrementais, onde depois alguns de seus autores se reuniram para discutir sobre essas metodologias e publicaram o manifesto ágil (BECK et al., 2001).

As metodologias ágeis seguem um conjunto de princípios e valores, como acolher mudanças, interação entre indivíduos, entrega constante de valor ao cliente, entre outros. Esses princípios e valores foram sumarizados no manifesto ágil (BECK et al., 2001), mas algumas dessas metodologias de desenvolvimento surgiram antes mesmo desse manifesto, como o Scrum em 1995 e o XP em 1996.

O Manifesto Ágil traz alguns valores que guiam o desenvolvimento visando melhorar a produção de software. Estes valores são:

- **Indivíduos e interações** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação de contratos;
- **Responder a mudanças** mais que seguir um plano.

O Manifesto Ágil reconhece o valor dos itens à direita, porém ele valoriza mais os itens à esquerda.

O Manifesto Ágil também traz 12 princípios que devem ser seguidos por seus praticantes:

1. A Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.

7. Software funcionando é a medida primária de progresso.

8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.

10. Simplicidade–a arte de maximizar a quantidade de trabalho que não precisou ser feito.

11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

As metodologias ágeis estão ganhando a atenção da indústria e está crescendo o número de praticantes dessas metodologias (RODRÍGUEZ et al., 2012).

2.2 Dívida Técnica

A Dívida Técnica (do inglês, *Technical Debt* - TD) é uma metáfora que faz alusões a eventos onde a qualidade do código é negligenciada a fim de se obter um benefício a curto prazo ou obter algum ganho competitivo (LIM; TAKSANDE; SEAMAN, 2012). Essa metáfora foi inicialmente proposta em (CUNNINGHAM, 1992), onde é dito que entregar o código negligenciado é como assumir uma dívida, sendo que um pouco de dívida acelera o desenvolvimento, desde que seja paga. O problema é que quando a dívida não é paga através da refatoração ela é acrescida de juros, que seria o aumento da complexidade do código e a degradação da qualidade do sistema (CUNNINGHAM, 1992)

A TD pode ser necessária em alguns contextos de negócio, como para acelerar o desenvolvimento de um produto para lançá-lo mais cedo no mercado e obter uma vantagem com relação aos concorrentes, ou para construir um protótipo funcional e garantir investimento do cliente no produto, como encontrado em (LIM; TAKSANDE; SEAMAN, 2012). Mas sua presença elevada pode tornar o software frágil, mais suscetível a *bugs*, o que prejudica o relacionamento com o cliente (LIM; TAKSANDE; SEAMAN, 2012) além de poder diminuir a produtividade do time (POWER, 2013).

Com o passar dos anos essa metáfora passou a ser mais amplamente utilizada, ganhando atenção tanto da indústria quanto da academia (LI; AVGERIOU; LIANG, 2015) devido a uma aproximação de linguagem que ela traz entre a área técnica e a área do negócio além da referência a degradação da qualidade do sistema. Com isso, a sua aplicação foi se estendendo a outros artefatos e fases do desenvolvimento de software além do código, de onde originou-se a metáfora (LI; AVGERIOU; LIANG, 2015).

Também foram caracterizados outros atributos para a metáfora em que os principais atributos são **Principal** e **Juros**. Em (OLIVEIRA, 2011) é apresentado um quadro compilando os diversos atributos e conceitos associados a TD encontrados em sua pesquisa, que são apresentadas na tabela 1 (OLIVEIRA, 2011).

Tabela 1 – Conceitos e atributos relacionadas a TD (Adaptado de (OLIVEIRA, 2011)

Conceito	**Definição**
Razão	Motivação que levou à decisão de comprometer a qualidade do sistema
Benefício	Consequências positivas da aquisição da dívida
Resultados	Consequências em longo prazo da contração da dívida
Principal	Valor que indica quanto custaria para pagar a dívida no momento
Juros	Possível penalidade em decorrência da dívida
Ganho ou valor rendido	Valor gerado em resultado dos benefícios alcançados
Rentabilidade	Relação de custo/benefício da dívida.

A Razão é o conceito relacionado ao que motivou a aquisição daquela dívida técnica, sendo alguns exemplos a restrição de capital, restrição de tempo (SEAMAN; GUO, 2011) e aproveitamento de oportunidades de mercado (LIM; TAKSANDE; SEAMAN,

2012). O Benefício é o conceito relacionados aos ganhos em curto prazo da aquisição da dívida, como aumento da produtividade (CUNNINGHAM, 1992) (SEAMAN; GUO, 2011). O Resultado diz respeito aos efeitos da TD a longo prazo, como dificuldade de manutenção do sistema e prejuízos na relação com o cliente (LIM; TAKSANDE; SEAMAN, 2012). A rentabilidade leva em conta o custo para pagar o principal e o juros e o valor ganho.

Dentre as discussões que existem a cerca da TD há uma discussão relacionada a se dívida técnica é apenas aquela perda de qualidade feita de forma intencional, a fim de alcançar um objetivo, ou se toda perda de qualidade caracteriza TD. Em (FOWLER, 2009) é argumentado que a TD é uma metáfora e a discussão deveria se centralizar se a metáfora ajuda ou não a entender como lidar com problemas de qualidade de código e como comunicá-los. Assim, mesmo com dívidas não intencionais a metáfora ainda ajuda tanto o time quanto a área de negócio a entender a qualidade atual do sistema. Além da intencionalidade, Fowler também discute sobre a prudência da TD, que reflete se o time está preocupado com os impactos daquela dívida, e não apenas assumindo ela de maneira negligente, classificando a TD em intencional/inconsciente e prudente/imprudente, formando assim um quadrante, que é ilustrado na figura 1.

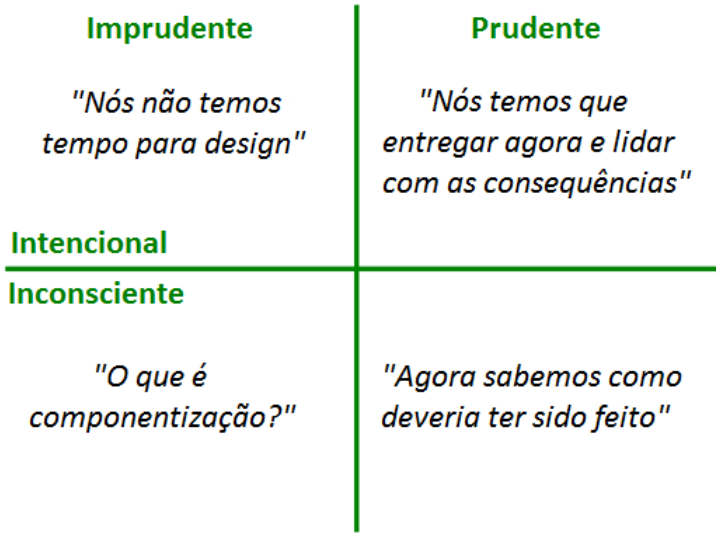

Figura 1 – Quadrante da Dívida Técnica (Fonte: (FOWLER, 2009) adaptado)

Em (JUNIOR, 2016) é explicado as situações da TD em cada um dos quadrantes:

1. **Imprudente e Intencional**: Situações onde a equipe decide utilizar soluções rápidas e de menor qualidade porque não possui tempo para construir algo de maior qualidade.
2. **Prudente e Intencional**: A equipe conhece as limitações impostas e decide/precisa entregar o software mesmo assim. Ela opta proativamente por tomar a TD após avaliar seus impactos e os seus benefícios.

3. **Imprudente e Inconsciente**: A equipe possui baixa qualidade técnica e não sabe projetar sistemas ou aplicar boas práticas e disso resulta uma codificação bagunçada e de menor escalabilidade e de difícil manutenção.

4. **Prudente e Inconsciente**: A princípio a equipe constrói um código que acredita ser de qualidade, mas após ganhar experiência ou informações relevantes para o contexto percebe que havia soluções melhores que poderiam ter sido construídas.

A TD sendo intencional ou não possui impactos nos projetos de software e por isso deve ser gerida (LIM; TAKSANDE; SEAMAN, 2012). Técnicas como a refatoração abordam diretamente a TD visando reduzir seu nível no projeto, mas antes mesmo disso a dívida tem que ser identificada. Além da identificação, a medição da dívida pode auxiliar gestores de software a tomar decisões relacionadas a TD de forma mais objetiva. Em (LI; AVGERIOU; LIANG, 2015) é realizado um mapeamento sistemático onde são identificadas 8 categorias de atividades de gestão da TD, sendo que as 3 primeiras atividades são as mais estudadas e propostas:

- Identificação - Atividades que detectam a TD através de alguma técnica, como análise estática de código.
- Medição - Quantifica o custo da TD conhecida em um software, através de técnicas de estimativas.
- Priorização - Classifica a TD identificada de acordo com regras predefinidas, estabelecendo uma prioridade entre classificações para decidir quais itens de TD devem ser pagos primeiros em relação a outros.
- Prevenção - Atividades que buscam prevenir uma potencial dívida de ser adquirida.
- Monitoramento - Atividades que buscam observar as mudanças do custo e dos benefícios da TD durante o tempo.
- Pagamento - Atividades que buscam resolver ou mitigar a TD através de técnicas de reengenharia e refatoração.
- Documentação - Atividades que buscam representar e codificar a TD de uma maneira uniforme tendo em mente preocupações de *stakeholders* específicos.
- Comunicação - Atividades que buscam tornar visível a TD identificada para *stakeholders* a fim de discutir como ela deverá ser gerenciada.

Com a identificação e a medição da TD o time ganha uma percepção de quanta dívida vem adquirindo, podendo aproveitar melhor ela (ALLMAN, 2012), e como ela vem

evoluindo com o tempo através de técnicas de monitoramento. A priorização da dívida permite aos gestores otimizar o tempo que vai ser gasto para se pagar a dívida, buscando uma relação saudável entre valor agregado entregue e manutenção da qualidade. Essa priorização pode ser feita em conjunta com outros *stakeholders* com auxílio das atividades de comunicação e documentação, e a dívida priorizada será saldada de acordo com as técnicas de pagamento definidas. As atividades de prevenção buscam evitar a aquisição de TD que não é estratégica para a organização.

2.2.1 Dívida Técnica nas Metodologias Ágeis de Desenvolvimento de Software

Recentemente a TD está se tornando um conceito popular dentro das metodologias ágeis de desenvolvimento de software (LETOUZEY, 2012) (BEHUTIYE et al., 2017), devido a suas implicações econômicas e aos indícios de que essas metodologias de desenvolvimento estão mais sujeitas a TD (BEHUTIYE et al., 2017). Dentre uma das principais causas para aquisição de TD temos a ênfase em entrega rápida de funcionalidades (TORKAR, 2011) (BAVANI, 2012) e problemas de arquitetura e design (TORKAR, 2011) (KTATA; LÉVESQUE, 2010). Com relação aos impactos da TD nessas metodologias, temos a redução da produtividade (BAVANI, 2012) (FERNáNDEZ-SáNCHEZ et al., 2014), a degradação da qualidade do sistema (TORKAR, 2011) (NORD et al., 2012), a degradação do relacionamento com o cliente (BAVANI, 2012), entre outros.

Contudo, o relacionamento das metodologias ágeis de desenvolvimento com a TD não é apenas negativo. Existem estudos indicando que essas metodologias podem utilizar a TD como estratégia para ganhar oportunidades de negócio em um mercado competitivo (NORD et al., 2012). Assim, entender a TD e seus impactos é importante para as metodologias ágeis (BEHUTIYE et al., 2017), começando pelo próprio conceito de TD. Dentre as percepções do que é TD encontrados na literatura em um contexto de metodologias ágeis temos "Consequências decorrentes de um desenvolvimento ruim de software"(KTATA; LÉVESQUE, 2010) (FERNáNDEZ-SáNCHEZ et al., 2014), "desvios de princípios de design e arquiteturais"(BAVANI, 2012) (TORKAR, 2011), "falta de conhecimento ou documentação inadequada"(CODABUX; WILLIAMS, 2013). Mas esses estudos também indicam que ainda falta uma concepção sólida de como caracterizar e gerenciar a TD nas metodologias ágeis de desenvolvimento (CODABUX; WILLIAMS, 2013).

2.2.2 Técnicas de Identificação, Medição e Comunicação da Dívida Técnica

Devido aos impactos da TD no desenvolvimento de software foram propostas maneiras de se gerenciar essa dívida a fim de potencializar os ganhos com a sua aplicação em um contexto de mercado, em que as atividades de gestão mais propostas foram as de identificação, medição e pagamento (LI; AVGERIOU; LIANG, 2015). Para atender ao

objetivo desse trabalho de **promover a visibilidade da TD** foi dado um enfoque na busca de técnicas que suportassem as atividades de identificação, medição e comunicação, que estão apresentadas mais abaixo.

Estimativa da TD através de fórmula: Em (CURTIS; SAPPIDI; SZYNKARSKI, 2012) os autores apresentam uma fórmula para estimar o valor do *principal* da TD de uma aplicação. Essa fórmula leva em consideração três variáveis: o número de violações que devem ser consertadas, a quantidade de horas para consertar cada violação e o custo da mão-de-obra. A variável de número de violações é justificada pelos autores alegando que as organizações possuem limitações de orçamento e não irão consertar todos os erros de código, e somente aqueles que serão consertados deverão ser levados em conta para a estimativa do principal. Além disso, a organização pode visar consertar uma porcentagem diferente das violações dependendo de seu nível de gravidade. A fórmula propostas pelos autores é apresentada a seguir:

$$
\begin{aligned}
\textit{Principal da TD} = & \Big((\sum \textit{Violações graves}) \times (\textit{Porcentagem a ser consertada}) \\
& \times (\textit{Média de horas para consertar}) \times (\textit{R\$ por hora}) \Big) + \\
& \Big((\sum \textit{Violações médias}) \times (\textit{Porcentagem a ser consertada}) \\
& \times (\textit{Média de horas para consertar}) \times (\textit{R\$ por hora}) \Big) + \\
& \Big((\sum \textit{Violações leves}) \times (\textit{Porcentagem a ser consertada}) \\
& \times (\textit{Média de horas para consertar}) \times (\textit{R\$ por hora}) \Big)
\end{aligned} \tag{2.1}
$$

Os autores ainda recomendam que as organizações que adotarem essa fórmula ajuste os parâmetros, como a porcentagem a ser consertada, o valor do preço por hora e o tempo, adicionar ou remover níveis de gravidade de violações, para atenderem a sua realidade. Um ponto importante em se observar é que a estimativa de horas trabalhadas para se consertar uma violação pode ser subjetiva, variando entre organizações dependendo de sua experiência.

É importante notar que a fórmula como está apresentada estimará apenas a quantidade de TD que a organização deseja mitigar, sendo que se ela deseja calcular toda a TD presente na aplicação ela deverá assumir como 100% as porcentagens a serem consertadas em todos os níveis.

Essa técnica pode ser classificada apenas como uma técnica de medição, pois ela já parte de um pressuposto de que a TD foi identificada anteriormente, servindo de insumo para a estimativa.

Método SQALE: O método SQALE (*Software Quality Assessment Based on*

Lifecycle Expectations) é um método de estimativa de dívida técnica proposto por (LETOUZEY, 2012) baseado em um modelo de qualidade e um modelo de análise, sendo que o primeiro traz a definição da organização de código correto, em forma de requisitos, e o segundo modelo define a estratégia para se consertar cada uma das violações aos requisitos, além do esforço necessário para se consertar essas violações.

O modelo de qualidade do SQALE é organizado em três níveis de hierarquia: Características, Subcaracterísticas e Requisitos, sendo o último as definições de código correto. Estes requisitos devem ser verificáveis, para que seja possível identificar uma violação aos mesmos. A tabela 2 ilustra um exemplo de requisitos do modelo de qualidade.

Tabela 2 – Exemplo de Modelo de Qualidade SQALE

Característica	**Sub-característica**	**Requisitos**
Instabilidade	Instabilidade relacionada a arquitetura	Não há dependência cíclica entre pacotes
Confiabilidade	Confiabilidade relacionada a dados	Não há comparações entre pontos flutuantes
Confiabilidade	Confiabilidade relacionada a cobertura	Todos arquivos possuem cobertura de testes superior a 70%
Testabilidade	Testabilidade em nível de teste unitário	Não há método com complexidade ciclomática acima de 12

Já o modelo de análise mapeia um requisito com uma remediação, isto é a tarefa a ser executada para normalizar uma violação aos requisitos do modelo de qualidade. O modelo de análise também associa cada remediação com uma função de remediação, que seria o esforço necessário para executar a remediação. A tabela 3 ilustra um exemplo de modelo de análise do SQALE.

Tabela 3 – Exemplo de Modelo de Análise SQALE

Requisitos	**Remediação**	**Função de Remediação**
Todos arquivos possuem cobertura de testes superior a 70%	Escrever testes adicionais	20 min por linha não coberta
Não há método com complexidade ciclomática acima de 12	Refatorar método até reduzir a complexidade ciclomática	30 min por ocorrência
Indentação de código seguindo uma regra consistente	Reparar indentação com ajuda da IDE	2 min por arquivo

Essa técnica pode ser classificada como de identificação e medição. Com o Modelo de Qualidade definido, é possível comparar o código com os requisitos definidos, e qualquer violação a esses requisitos é caracterizado dívida técnica. Essa comparação pode ser apoiada por ferramentas de análise estática. Com as violações identificadas, é possível utilizar o modelo de análise para calcular o esforço necessário para remediar todas as violações, tendo ao final uma estimativa da TD em termos de esforço.

Comentários de código (dívida técnica admitida): A dívida técnica admitida é a TD que é adquirida intencionalmente e registrada explicitamente no código através de comentários (BAVOTA; RUSSO, 2016). Existem técnicas que buscam minerar o código

e identificar comentários que indiquem a TD. Em (FARIAS et al., 2015) é apresentado um modelo, chamado de CVM (do inglês, *Contextualized Vocabulary Model*), para poder identificar esses tipos de comentários, onde é realizado um mapeamento de um conjunto de palavras, como substantivos, advérbios, verbos, etc., e tenta relacioná-los com tipos de TD. Os autores apontam alguns benefícios do modelo, como o vocabulário que é aplicado no modelo serve para identificar a TD, mas também concluem que o modelo precisa ser melhorado para diferenciar comentários comuns de comentários que indiquem realmente a TD.

Essa técnica é classificada como de Identificação somente.

Identificação Manual: Em (ZAZWORKA et al., 2013) é realizado um comparativo entre uma identificação manual de TD com a identificação realizada por três ferramentas de análise estática de código (FindBugs, Code Smells, Metrics). A justificativa dos autores para esse comparativo é porque a identificação humana consegue pegar tipos de dívidas que essas ferramentas não conseguem, como de documentação e usabilidade, além de que as ferramentas analisadas só capturaram metade das dívidas identificadas de *design*, justificando assim os benefícios de se aliar a identificação manual com estratégias automatizadas.

Essa identificação manual é realizada através de um *template* que é preenchido para registrar a dívida. No estudo conduzido foram utilizados diferentes papéis para fazer a realização da identificação, sendo 2 desenvolvedores, um "mantedor", um *tester*, e um gerente. Esses diferentes papéis capturaram diferentes tipos de dívidas, ressaltando a importância de se ter essa diversidade na hora de identificar a TD.

Essa TD então é ilustrada através de um cubo em visão isométrica onde se consegue ver apenas três faces, como mostra a Fig. 2. Cada uma dessas faces possui uma cor e representa um atributo da dívida, sendo eles o Principal, Juros e Probabilidade do juros. As cores variam entre verde, amarelo e vermelho, sendo uma mais grave que a outra nessa ordem. Para o principal, quanto mais grave a cor, maior é o esforço para se reparar a dívida; para os juros, quanto mais grave a cor, maior será o retrabalho, caso a dívida não seja paga; para a probabilidade dos juros, quanto mais grave a cor, maior será a chance de haver a necessidade do retrabalho representado pela face deles.

Essa técnica é classificada pode ser classificada como de identificação, documentação e comunicação, pois ela a TD é identificada e registrada através dessa técnica, provendo assim uma ilustração da TD presente no código. Ela não pode ser considerada como de medição pois apesar de haver uma representação do principal e do juros essa representação é muito subjetiva e não traz valores quantitativos.

Code Christmas Tree: Em (KAISER; ROYSE, 2011) é apresentada uma técnica para tornar a TD visível a todos. Consiste em um quadro onde há vários retângulos

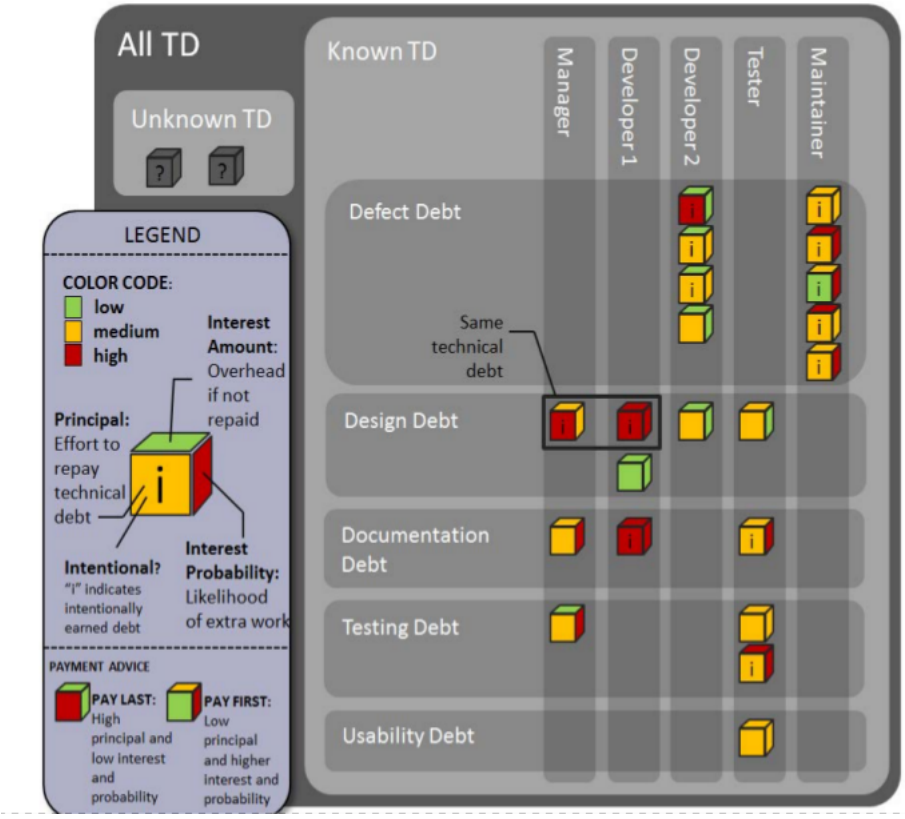

Figura 2 – *Resultados da elicitação manual (fonte: (ZAZWORKA et al., 2013))*

com cores que variam do verde ao vermelho para ilustrar algum aspecto de qualidade das classes/arquivos de código, como cobertura de testes, complexidade ciclomática, etc., como ilustrado na Fig. 3 . Essa representação gráfica deve estar em um local visível e de preferência onde todos os interessados tenham acesso. Com ela é possível rapidamente perceber classes/arquivos que possuem uma densidade alta de dívida técnica, e isto conscientiza os desenvolvedores.

Em (KAISER; ROYSE, 2011) também é relatado como benefício a percepção por parte de outros envolvidos fora do time do desenvolvimento, como a área de negócio ou o cliente. Uma vez que o quadro está explícito para todos, os demais *stakeholders* conseguem perceber a evolução da qualidade do código quando os retângulos vermelhos/alaranjados se tornam verde-claro/verde-escuro. Com isso é mais fácil justificar o esforço em melhorar a qualidade do código, pois o cliente agora consegue ter uma percepção dos efeitos desse esforço.

Em (LI; AVGERIOU; LIANG, 2015) essa técnica é classificada como sendo de Comunicação.

Figura 3 – *Code Christmas Tree* fonte: (KAISER; ROYSE, 2011)

2.3 Considerações Finais

Neste capítulo foram apresentados os princípios e valores das metodologias ágeis e os principais conceitos relacionados a dívida técnica, além de seus impactos nas metodologias ágeis bem como algumas técnicas propostas para realização da sua gestão. É possível perceber que as metodologias ágeis buscam a entrega contínua de valor ao cliente e que isso deve ser alcançado mantendo uma boa qualidade do código escrito, pois isso aumenta a agilidade (BECK et al., 2001).

Assim, a TD é um tema de interesse para essas metodologias pois ela retrata justamente a qualidade do código, que pode impactar na agilidade do time. Essa entrega com qualidade pode ser difícil em um contexto de mercado com restrições de custo e tempo, e a gestão da TD se faz importante para conseguir conquistar oportunidades de mercado enquanto mantém a qualidade do sistema, tornando necessário o emprego de técnicas para realizar esta gestão.

3 Método de Promoção da Visibilidade e Métrica da Dívida Técnica

Retomando o objetivo proposto para este trabalho de **construir um método que apoie a promoção da visibilidade da TD**, foi realizada uma pesquisa na literatura para encontrar quais técnicas existem de identificação, medição e comunicação da TD, que são as atividades TDM que estão mais relacionadas com a visibilidade da TD, que foram apresentadas no capítulo passado. Neste capítulo será discutido como essas técnicas podem ser aplicadas a fim de constituir um método que permita a identificação, medição e comunicação da TD, além de uma discussão sobre as métricas que algumas dessas técnicas utilizam, para poder ser debatido se elas são viáveis para um contexto de metodologias ágeis.

3.1 Método e Métrica

3.1.1 Técnicas para compor o método

3.1.1.1 Seleção das técnicas

Foram apresentadas 5 técnicas, sendo que 3 suportavam atividade de **Identificação**, 2 suportavam atividade de **Medição**, e 1 atividade de **Comunicação**, sendo que uma técnica pode suportar mais de uma atividade ao mesmo tempo. Para poder selecionar quais delas serão adotadas para compor o método a ser apresentada neste trabalho, foi construído um quadro comparando as técnicas com alguns atributos desejáveis a fim de selecionar a técnica que melhor atendesse esses atributos.

Os atributos definidos foram divididos entre atributos essenciais e atributos diferenciais, sendo que o método necessariamente terá que atender todos os atributos essenciais, enquanto os atributos diferenciais agregam para o método, mas não são críticos. Os atributos essenciais são aqueles referentes às técnicas que suportam as atividades TDM de interesse, que são identificação, medição e comunicação. O atributo diferencial é a de automação da técnica através de uma ferramenta, o que facilita o emprego da técnica em um contexto de desenvolvimento

A baixo estão os atributos adotados para a seleção das técnicas:

1. **Suporta atividades de Identificação (TDM - Identificação)**: Atributo essencial;
2. **Suporta atividades de Medição (TDM - Medição**: Atributo essencial;

3. **Suporta atividades de Comunicação (TDM - Comunicação)**: Atributo essencial;

4. **Técnica pode ser automatizada por ferramenta**: Atributo diferencial;

Assim, a tabela a seguir foi construída comparando as técnicas com esses atributos.

Tabela 4 – Comparação das técnicas com os atributos de seleção

Técnicas	**TDM - Identificação**	**TDM - Medição**	**TDM - Comunicação**	**Pode ser automatizada**
Estimativa através de fórmulas		X		X
SQALE	X	X		X
CVM	X			X
Identificação Manual	X			
Code Christmas Tree			X	

Através da tabela 4 é possível perceber que nenhuma técnica sozinha contempla todos os atributos essenciais, indicando a necessidade do método compor mais de uma técnica para poder satisfazer o objetivo proposto. Contudo, durante a busca de ferramentas que implementassem alguma dessas técnicas, foi encontrado em (LI; AVGERIOU; LIANG, 2015) uma lista de ferramentas que implementavam atividades TDM. Nessa lista havia um *plugin* para a ferramenta *SonarQube* que implementava o método SQALE de maneira automatizada, sendo que esses autores classificaram o *plugin* como atividade de identificação, medição e comunicação.

Ao realizar uma busca pela ferramenta e o *plugin*, foi identificado que na versão atual do *SonarQube* ele já traz a análise de TD através do método SQALE sem a necessidade do *plugin*. Assim, a análise de TD feita pelo *SonarQube* utilizando o SQALE contemplaria os 3 atributos essenciais, além de também contemplar o atributo desejável, que seria a automação.

Nas figuras 4, 5 e 6 são ilustradas como a ferramenta *SonarQube* comunica a TD estimada, como o gráfico da fig. 5 em que este possui dois eixos, linhas de código e dívida técnica, onde as classes são representadas por bolhas de tamanho e cor variáveis, sendo que esta representa a proporção de TD, que é a relação entre o custo para desenvolver software e o custo para consertá-lo (S.A, 2018b), e aquele representa a quantidade de *code smells* presente na classe.

3.1.1.2 Composição do SQALE com a estimativa através de fórmula

Durante as análises das técnicas apresentadas, e levando em conta a discussão sobre as vantagens das métricas em **esforço** e em **valores financeiros**, foi percebido a possibilidade de utilizar o SQALE junto com a técnica de estimativa através de fórmula

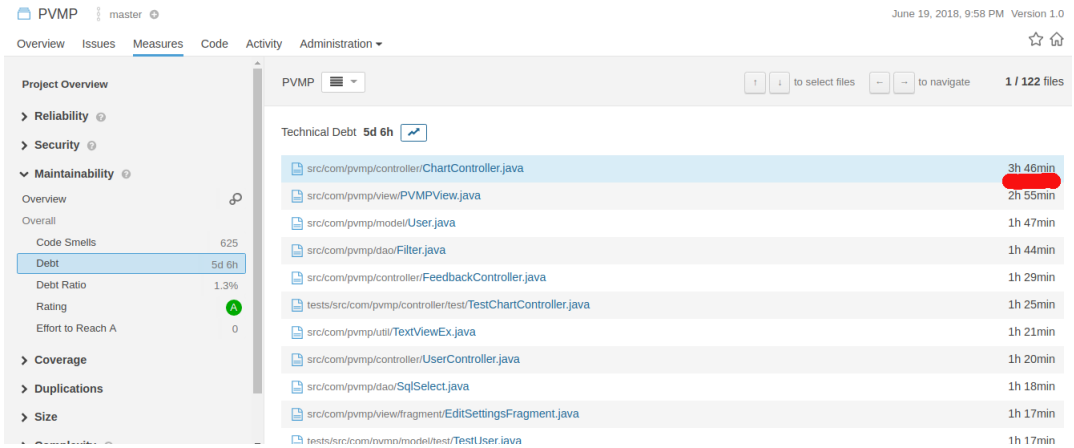

Figura 4 – Estimativa de Dívida Técnica dada pela ferramenta SonarQube

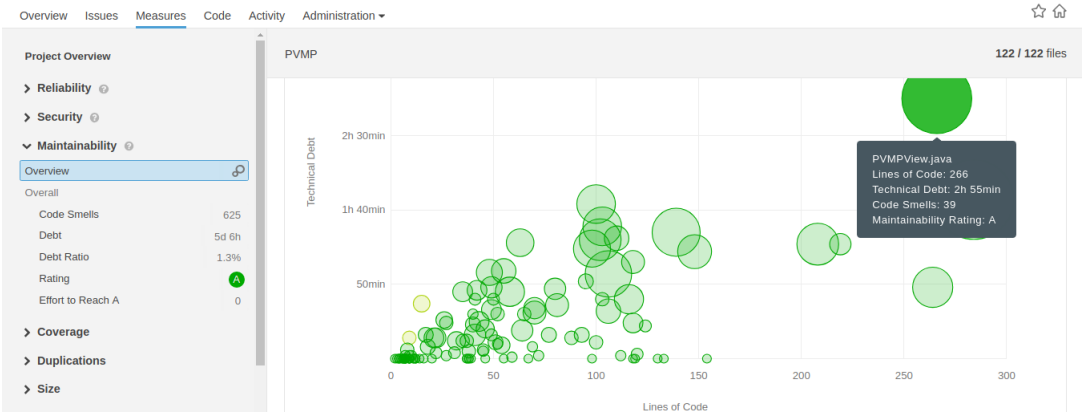

Figura 5 – Ilustração da TD presente em um projeto através de um gráfico

para poder realizar uma tradução da métrica em esforço para a métrica em valor financeiro. Essa tradução pode ser realizada através da substituição dos termos em vermelho da equação 3.1 pela estimativa produzida pelo SQALE:

$$\begin{aligned} \textit{Principal da TD} = & \Big((\sum \textit{Violações graves}) \times (\textit{Porcentagem a ser consertada}) \\ & \times (\textit{Média de horas para consertar}) \times (\textit{R\$ por hora}) \Big) + \\ & \Big((\sum \textit{Violações médias}) \times (\textit{Porcentagem a ser consertada}) \\ & \times (\textit{Média de horas para consertar}) \times (\textit{R\$ por hora}) \Big) + \\ & \Big((\sum \textit{Violações leves}) \times (\textit{Porcentagem a ser consertada}) \\ & \times (\textit{Média de horas para consertar}) \times (\textit{R\$ por hora}) \Big) \end{aligned} \tag{3.1}$$

Resultando em:

$$\textbf{Principal da TD} = \textit{Estimativa do SQALE} \times \textit{R\$ por hora.} \tag{3.2}$$

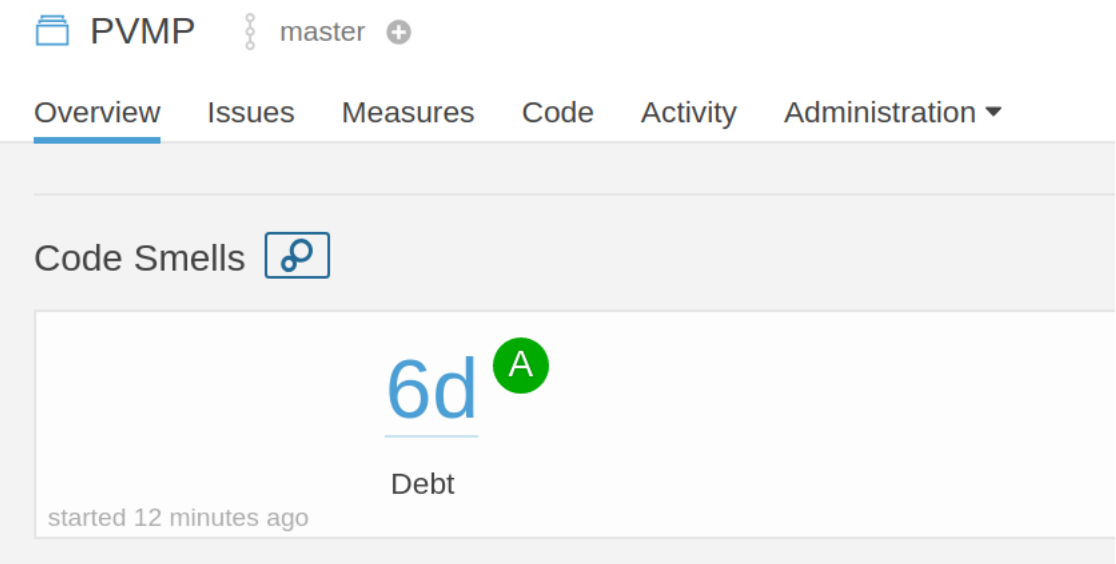

Figura 6 – Representação do valor total da TD presente em um projeto

Assumindo que a estimativa foi dada em horas. Caso seja outra unidade ou deverá haver a conversão da estimativa para horas ou a conversão do valor da mão-de-obra para a unidade da estimativa.

O raciocínio por trás dessa adaptação se deu através da observação de que os termos em vermelho na equação 3.1 refletem um número dado em termos de esforço, que seria a média de tempo necessária para saldar a TD que se pretende saldar de um determinado nível de gravidade de violação. Contudo, o nosso método busca representar toda a TD e não apenas o que se pretende saldar, sendo assim a separação por níveis de gravidade se faz desnecessária. O que sobra então seria um somatório total do esforço de todas as violações, que é o que o SQALE nos traz. Assim, podemos substituir esses termos pela estimativa do SQALE evitando a coleta manual desses valores e conseguindo traduzir a métrica de esforço trazida pelo SQALE para uma métrica de valores financeiros com um produto entre a estimativa do SQALE e o preço da mão de obra. Essa adaptação simplificada é mais interessante em um contexto de metodologias ágeis.

3.1.2 Métricas utilizadas para medição da TD

A TD foi definida originalmente por (CUNNINGHAM, 1992) como a perda de qualidade do código. Tendo em vista a alusão financeira da metáfora algumas técnicas de estimativas trazem valores financeiros para representar a dívida como a técnica de estimativa da fórmula 2.1 proposta por (CURTIS; SAPPIDI; SZYNKARSKI, 2012). Contudo, algumas outras técnicas colocam essa estimativa em termos de esforço necessário para reparar essa perda de qualidade sendo representado por valores de tempo, como minutos, horas ou até dias, como o método SQALE (LETOUZEY, 2012).

É interessante notar que para a estimativa através da fórmula 2.1 é necessário um insumo em termos de tempo para obter o resultado financeiro. Assim, a representação da TD através de um esforço de tempo pode ser traduzida para um custo financeiro fazendo uma multiplicação entre o tempo necessário para pagar a dívida e o custo de mão-de-obra daqueles que irão pagá-la. As duas representações da TD possuem características interessantes de um ponto de vista de gestão, pois a representação em termos de esforço auxilia na tomada de decisão relacionadas a cronograma, permitindo uma alocação mais estratégica de recursos humanos para desenvolver novas funcionalidades e/ou dar manutenção na qualidade do código. Já a representação financeira é mais adequada para tomada de decisões que levem em conta o orçamento, ou para ter uma representação da qualidade do código em um parâmetro mais familiar para área de negócio.

3.1.3 Comunicação da TD estimada

Apesar da ferramenta *SonarQube* suportar as três atividades de gestão de TD, identificação, medição e comunicação, como a comunicação é realizada diretamente pela ferramenta a TD só ficaria visível para aqueles que lidasse com ela diretamente, sendo necessária a pró-atividade dos interessados em querer ir lá averiguar os níveis da TD.

Para garantir uma comunicação mais efetiva, a técnica do *Code Christmas Tree* pode ser utilizada para representar os valores estimados da TD, sendo eles ou em esforço ou em valores financeiros, colocando-se o quadro em um ambiente onde os interessados estejam presentes ou passem por ele de preferência diariamente.

É importante apontar que a construção e manutenção manual do quadro não é muito interessante pois precisa de muito tempo para ficar atualizando o quadro. Contudo, o gráfico que o quadro representa pode ser implementada por programa e ser ilustrado em uma televisão por exemplo, facilitando sua manutenção e podendo até ser automatizada.

3.1.4 Proposta de método

Com as técnicas escolhidas, além de uma discussão sobre a possibilidade de composição de técnicas para flexibilização da métrica utilizada, foi composto um método para a promoção da visibilidade da TD. O método proposto utiliza o SQALE através da ferramenta *SonarQube*, que permite a **identificação**, **medição** e **comunicação** da TD além de ser automatizada, juntamente com o *Code Christmas Tree* para fortalecer a atividade de comunicação.

A estimativa de TD fornecido por essa ferramenta é dada em esforço sendo unidade de tempo a ser trabalhado, como minutos, horas ou até dias, e é representada diretamente na ferramenta permitindo a sua visibilidade para quem a utiliza. Essa estimativa também pode ser utilizada como insumo para a fórmula 3.2, traduzindo essa estimativa para valores

monetários, podendo fornecer assim as duas métricas. As duas estimativas então são utilizadas como parâmetro de representação dentro do quadro do *Code Christmas Tree* que deverá estar em um local visível para as partes interessadas. O método então possui duas possibilidades de representação da TD, possuindo um fluxo como o representado na fig. 7.

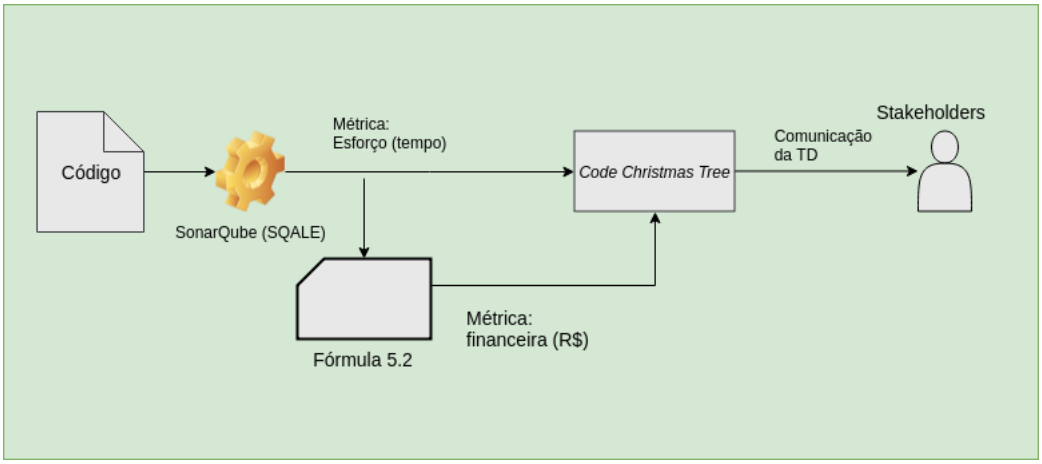

Figura 7 – Ilustração do método proposto (fonte: o autor)

3.2 Considerações Finais

Através das discussões sobre as métricas encontradas para se estimar a TD e os benefícios de cada uma de acordo com o ponto de vista de gestão e a análise realizada para a escolha das técnicas mais adequadas para compor o método, foi proposto um método que utiliza a ferramenta de análise de código *SonarQube*, que gera uma estimativa baseada no SQALE da TD presente no código. Além disso, a solução também conta com a fórmula 3.2 para traduzir a estimativa da ferramenta de esforço para valores monetários, trazendo assim uma flexibilização na métrica utilizada para servir de insumo para tomada de decisões levando em consideração diferentes pontos de vista.

4 Validação

Nesse trabalho foi discutido o problema da aquisição de TD durante o desenvolvimento de software em um contexto de metodologias ágeis, oriunda da ênfase em entrega rápida de funcionalidades, fator que foi identificado nesses times como sendo fonte de TD (BEHUTIYE et al., 2017). Uma das causas identificadas para esse problema foi a falta de visibilidade da TD para as partes interessadas.

Para mitigar essa causa foi realizada uma busca na literatura de métricas para estimativas de TD e de técnicas que permitissem a identificação, medição e comunicação da TD. Um método foi proposto compondo duas técnicas, que seria o SQALE através da ferramenta *SonarQube* e a técnica de estimativas por fórmula, que foi adaptada para a fórmula 3.2 e utilizando métricas de esforço ou valores monetários. Neste capítulo será discutido sobre a validação do método proposto, bem como suas limitações.

4.1 Validação do método

Para validar o método proposto este deverá ser aplicado em um contexto de desenvolvimento e deverá ser verificado se:

- O método realmente promove a visibilidade da TD, e
- Se a promoção da visibilidade realmente contribui para diminuição da aquisição da TD.

Contudo o método não pode ser aplicado durante o desenvolvimento desse trabalho, pois não houve disponibilidade de um ambiente adequado para aplicação da validação. Assim, será discutido se o método proposto possui evidências de promoção de visibilidade através do que é discutido na literatura, e se o método é viável para ser aplicado e validado em um contexto de desenvolvimento

4.1.1 Evidências Sobre a Promoção da Visibilidade

Tendo em vista que uma das questões de pesquisa desse trabalho é "*como dar visibilidade a TD*", tem que ser identificado quais atividades de gestão de TD promovem a visibilidade da TD. Das categorias de atividades TDM apresentadas em (LI; AVGERIOU; LIANG, 2015) 3 lidam diretamente com a visibilidade, que são as atividades de identificação, medição e comunicação. A identificação encontra onde a TD está presente

no código e a medição quantifica essa dívida, fornecendo o objeto que precisa ser visualizado. Contudo, não adianta ter esse valor coletado ou estimado se não há a divulgação desse valor para as partes interessadas, sendo este o papel das atividades de comunicação.

Logo, para termos um apoio teórico sobre se a solução promove ou não a visibilidade da TD, essa solução precisa suportar essas 3 atividades de gestão. Em (LI; AVGERIOU; LIANG, 2015) a ferramenta *SonarQube*, que é utilizada no método proposto, é classificada como uma ferramenta que dá suporte as atividades de identificação, medição e comunicação. A identificação e medição se dá através da implementação do SQALE pela ferramenta, e a comunicação é realizada pela própria ferramenta com a ilustração da TD presente no código através da representação em círculos, que variam de tamanho de acordo com o tamanho da classe e de cor de acordo com a densidade de TD.

4.1.2 Aplicação do Método para Validação

O método proposto tem sua operacionalidade ilustrada na fig. 7. Sua aplicação utiliza a ferramenta *SonarQube*, que é uma ferramenta de análise estática de código utilizada por mais de 85 mil organizações (S.A, 2018a). Isso demonstra que a ferramenta possui confiança da indústria e que é adequada para utilização durante o desenvolvimento, obtendo assim a estimativa em esforço do SQALE.

Para a estimativa financeira, o cálculo pode ser realizada de maneira manual após a coleta automatizada do *SonarQube*. Não foi analisado se a ferramenta permite a exportação dos dados analisados para utilização por um script externo para a automatização da fórmula 3.2. Contudo, ainda é viável a operacionalização do fluxo do método proposto para geração de estimativas em valores financeiros, visto que a coleta do esforço é automatizada e a conversão em valor financeiro é um produto simples.

4.2 Limitações

O método proposto neste trabalho tem por objetivo promover a visibilidade da TD para que as organizações percebam qual o estado atual da dívida em seus projetos e como eles estão evoluindo com o tempo, para permitir um desenvolvimento mais equilibrado com relação a entrega de novas funcionalidades e manutenção da qualidade do código.

Contudo, para realmente validar se o método promove os benefícios esperados este deveria ser aplicado em um contexto real de desenvolvimento para melhor se conhecer a extensão de seus benefícios e os eventuais problemas operacionais que poderiam vir a emergir. A validação apresentada neste trabalho não traz resultados quantitativos e qualitativos que permitam fazer uma análise com relação a esses pontos apresentados.

Outro ponto importante de limitação que deve ser apontado é com relação a mé-

trica de esforço e, consequentemente, a métrica de valor financeiro que depende da métrica de esforço. Essa estimativa de esforço é obtida através da ferramenta *SonarQube* que estima utilizando o SQALE. Contudo, o modelo de análise do SQALE, que é onde são definidos os índices de remediação para cada violação, não pode ser editado diretamente na ferramenta, sendo que a estimativa utiliza valores padrões que não necessariamente corresponderão ao esforço necessário para a organização saldar aquela dívida estimada. Uma base história seria necessária para calcular desvios padrões das estimativas.

Além disso, é importante perceber que em uma jornada de trabalho de 8 horas/dia as 8 horas trabalhadas não são todas produtivas. Assim, nas tomadas de decisões é importante levar em conta a cultura organizacional e a produtividade efetiva dos membros da equipe para se ter noção de quanto tempo realmente será necessário para saldar a TD.

5 Considerações Finais

Este trabalho possui como objetivo a proposta de um método para a promoção da visibilidade da TD com o objetivo de se mitigar a aquisição de TD nas metodologias ágeis devido a uma ênfase em entrega rápida de funcionalidades. Para alcançar esse objetivo foi realizado uma busca na literatura sobre quais técnicas existem que suportem as atividades de identificação, medição e comunicação da TD, sendo que essas três atividades são as mais relacionadas com a visibilidade da TD. Com a identificação dessas técnicas também foram encontradas quais métricas elas utilizam para representar a TD, sendo as métricas encontradas a de esforço e valores financeiros.

A partir de critérios propostos, foi escolhida uma métrica para representação da TD, que foi a medida em **esforço**, e foram selecionados técnicas para compor a estratégia. Contudo, foi percebido que ao compor duas das técnicas encontradas, que foi a análise pelo método *SQALE* através da ferramenta *SonarQube*, e a fórmula proposta por (CURTIS; SAPPIDI; SZYNKARSKI, 2012), seria possível realizar uma tradução da estimativa da TD em esforço para uma estimativa financeira. Assim, apesar de a métrica em esforço ser a métrica escolhida, a estratégia proposta seria capaz de fornecer estimativas nas duas métricas, dando flexibilidade e podendo aproveitar os benefícios de cada um dos tipos de representação.

Para validar a estratégia, foi buscado na literatura quais atividades promovem a visibilidade da TD e verificado se a estratégia proposta contempla essas atividades, para se obter indícios dessa promoção da visibilidade. Foi possível confirmar de que a estratégia possui indícios de que promoveria a visibilidade.

Contudo, a estratégia não pode ser aplicada em um contexto para uma análise mais rigorosa. Apesar do indício teórico de que a estratégia promoveria a visibilidade, ter uma análise aplicada da estratégia em um contexto de desenvolvimento poderia confirmar melhor essa hipótese, além de evidenciar possíveis limitações e oportunidades de melhoria da estratégia.

Outro ponto importante de se ressaltar é que, apesar da estratégia promover a visibilidade da TD, não se sabe quais os impactos positivos dessa promoção da visibilidade na mitigação da aquisição de TD pelas metodologias ágeis devido a ênfase na entrega de funcionalidades. Ao promover a visibilidade da TD em uma equipe que desenvolve seguindo as metodologias ágeis, como elas utilizariam essa informação para gerir a TD? As métricas escolhidas são adequadas para que os *stakeholders* entendessem os impactos da TD no desenvolvimento? Essas são questões de relevância que não podem ser respondidas apenas com uma análise na literatura sobre se a solução promove ou não a visibilidade

da TD.

Assim, uma proposta de trabalhos futuros seria a aplicação dessa estratégia em um contexto, através de técnicas como Estudos de Caso ou experimentação. Isso permitiria conhecer melhor os limites da estratégia e observar como os seus impactos positivos podem ou não ajudar na gestão da TD.

Referências

ALLMAN, E. Managing technical debt. *Commun. ACM*, ACM, New York, NY, USA, v. 55, n. 5, p. 50–55, maio 2012. ISSN 0001-0782. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/2160718.2160733>. Citado 2 vezes nas páginas 22 e 29.

BAVANI, R. Distributed agile, agile testing, and technical debt. *IEEE Software*, v. 29, n. 6, p. 28–33, Nov 2012. ISSN 0740-7459. Citado na página 30.

BAVOTA, G.; RUSSO, B. A large-scale empirical study on self-admitted technical debt. In: *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. [S.l.: s.n.], 2016. p. 315–326. Citado na página 32.

BECK, K. et al. *Manifesto para Desenvolvimento Ágil de Software*. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 06 out. 2017. Citado 2 vezes nas páginas 25 e 35.

BEHUTIYE, W. N. et al. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology*, v. 82, p. 139–158, fev. 2017. ISSN 09505849. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0950584916302890>. Citado 4 vezes nas páginas 21, 22, 30 e 43.

CODABUX, Z.; WILLIAMS, B. Managing technical debt: An industrial case study. In: *2013 4th International Workshop on Managing Technical Debt (MTD)*. [S.l.: s.n.], 2013. p. 8–15. Citado 2 vezes nas páginas 22 e 30.

CUNNINGHAM, W. The wycash portfolio management system. In: *Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum)*. New York, NY, USA: ACM, 1992. (OOPSLA '92), p. 29–30. ISBN 0-89791-610-7. Disponível em: <http://doi.acm.org.ez54.periodicos.capes.gov.br/10.1145/157709.157715>. Citado 4 vezes nas páginas 21, 27, 28 e 40.

CURTIS, B.; SAPPIDI, J.; SZYNKARSKI, A. Estimating the principal of an application's technical debt. *IEEE Software*, v. 29, n. 6, p. 34–42, Nov 2012. ISSN 0740-7459. Citado 3 vezes nas páginas 31, 40 e 47.

FARIAS, M. A. d. F. et al. A contextualized vocabulary model for identifying technical debt on code comments. In: *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*. [S.l.: s.n.], 2015. p. 25–32. Citado na página 33.

FERNáNDEZ-SáNCHEZ, C. et al. Guiding flexibility investment in agile architecting. In: *2014 47th Hawaii International Conference on System Sciences*. [S.l.: s.n.], 2014. p. 4807–4816. ISSN 1530-1605. Citado na página 30.

FOWLER, M. *Technical Debt Quadrant*. 2009. Disponível em: <https://martinfowler.com/bliki/TechnicalDebtQuadrant.html>. Acesso em: 13 out. 2017. Citado 2 vezes nas páginas 11 e 28.

JUNIOR, M. M. B. *Estratégias de redução de Dívidas Técnicas em Equipes Ágeis*. São Paulo: [s.n.], 2016. Citado na página 28.

KAISER, M.; ROYSE, G. Selling the investment to pay down technical debt: The code christmas tree. In: *2011 Agile Conference*. [S.l.: s.n.], 2011. p. 175–180. Citado 4 vezes nas páginas 11, 33, 34 e 35.

KTATA, O.; LÉVESQUE, G. Designing and implementing a measurement program for scrum teams: What do agile developers really need and want? In: *Proceedings of the Third C* Conference on Computer Science and Software Engineering*. New York, NY, USA: ACM, 2010. (C3S2E '10), p. 101–107. ISBN 978-1-60558-901-5. Disponível em: <http://doi.acm.org/10.1145/1822327.1822341>. Citado na página 30.

LETOUZEY, J. L. The sqale method for evaluating technical debt. In: *2012 Third International Workshop on Managing Technical Debt (MTD)*. [S.l.: s.n.], 2012. p. 31–36. Citado 3 vezes nas páginas 30, 32 e 40.

LI, Z.; AVGERIOU, P.; LIANG, P. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, v. 101, p. 193 – 220, 2015. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121214002854>. Citado 8 vezes nas páginas 21, 27, 29, 30, 34, 38, 43 e 44.

LIM, E.; TAKSANDE, N.; SEAMAN, C. A balancing act: What software practitioners have to say about technical debt. *IEEE Software*, v. 29, n. 6, p. 22–27, Nov 2012. ISSN 0740-7459. Citado 5 vezes nas páginas 21, 22, 27, 28 e 29.

NORD, R. L. et al. In search of a metric for managing architectural technical debt. In: *Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*. Washington, DC, USA: IEEE Computer Society, 2012. (WICSA-ECSA '12), p. 91–100. ISBN 978-0-7695-4827-2. Disponível em: <http://dx.doi.org/10.1109/WICSA-ECSA.212.17>. Citado 2 vezes nas páginas 22 e 30.

OLIVEIRA, R. G. de. *Caracterização e Conceituação Teórica da Metáfora de Débito Técnico Através de um Estudo Exploratório*. Recife: [s.n.], 2011. Citado 3 vezes nas páginas 13, 21 e 27.

POWER, K. Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options. In: *Proceedings of the 4th International Workshop on Managing Technical Debt*. Piscataway, NJ, USA: IEEE Press, 2013. (MTD '13), p. 28–31. ISBN 978-1-4673-6443-0. Disponível em: <http://dl.acm.org/citation.cfm?id=2663297.2663302>. Citado 2 vezes nas páginas 21 e 27.

RODRÍGUEZ, P. et al. Survey on agile and lean usage in finnish software industry. In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: ACM, 2012. (ESEM '12), p. 139–148. ISBN 978-1-4503-1056-7. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/2372251.2372275>. Citado na página 26.

S.A, S. *Continuos Inspection | SonarQube*. 2018. Disponível em: <https://www.sonarqube.org/>. Citado na página 44.

S.A, S. *Metric Definitions*. 2018. Disponível em: <https://docs.sonarqube.org/display/SONAR/Metric+Definitions>. Citado na página 38.

SEAMAN, C.; GUO, Y. Chapter 2 - measuring and monitoring technical debt. In: ZELKOWITZ, M. V. (Ed.). Elsevier, 2011, (Advances in Computers, v. 82). p. 25 – 46. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123855121000025>. Citado 2 vezes nas páginas 27 e 28.

TORKAR, R. *Adopting Free/Libre/Open Source Software Practices, Techniques and Methods for Industrial Use*. Dissertação (Mestrado) — University of Gothenburg, 2011. Citado 2 vezes nas páginas 22 e 30.

ZAZWORKA, N. et al. A case study on effectively identifying technical debt. In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*. New York, NY, USA: ACM, 2013. (EASE '13), p. 42–47. ISBN 978-1-4503-1848-8. Disponível em: <http://doi.acm.org/10.1145/2460999.2461005>. Citado 3 vezes nas páginas 11, 33 e 34.